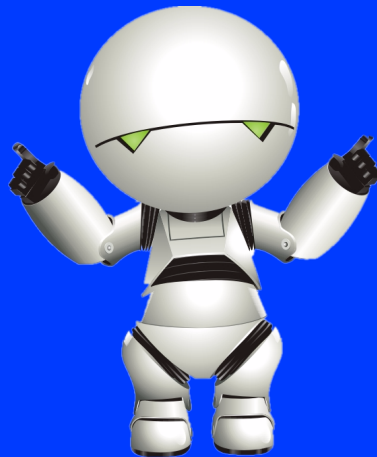


{EPITECH}

MY_MARVIN

< BECOME A JENKINS MASTER! />



MY_MARVIN

Compiling is fun (yes, please agree!), but compiling every time to see if nothing broke is not.

Testing a program is fun (yes, please agree again!), but forgetting to do it the one time you did a breaking change right before the delivery deadline is not.

Doing the same thing over and over again can be prone to errors and forgetting.



But a saviour is approaching: automation!

Automation technologies allow you to industrialize a lot of tasks, exactly how you want them to be executed.

During this bootstrap, you are going to become familiar with one of the most popular automation platform: *Jenkins*, and two of its associated tools *Configuration as Code* and *Job DSL*.



Jenkins enters the game!

A little quiz to start with

Before starting to use Jenkins, it is important to gain some knowledge and to either acquire or clarify important points. Try to answer those questions with your friends:

- ✓ What is Jenkins?
- ✓ What are Jenkins' jobs?
- ✓ Why using Jenkins instead of just bare shell scripts?
- ✓ Why is Marvin nicknamed the "Paranoid Android" and why do you have to answer this question anyway?

If you answered those questions right, you then have all you need to start your journey.

Setup time

To use Jenkins, you have to install it (natuurlijk ook).

There are several ways to install Jenkins, but using your sacred knowledge from your first DevOps project is by far the best way to proceed.

Not convinced? What if you discover that there exists official pre-built instances just waiting for you to be deployed? Well, thanks the whale. ;)



The version of Jenkins that the Automated Tests will use is the latest LTS one.



Time to automate

Tinker with your new friend and try to do the following tasks directly on your Jenkins instance.

System message

Your instance is not very welcoming, this is sad.

Set the system message to "Hello! Welcome to my Jenkins instance :)".

A visitor

Being anonymous is not fun, one should have an account with a great name! Create a user *Jigglypuff* with an id `jigglypuff`. Try to log in with it!

Your first job

Knowing the disk space usage in one click would be great. Create a freestyle job named "Disk Space Check" which executes the shell command `df`.



"This desk job is killing me Jim. I need to be out in the field."

A polite DDoS

Having a bad day? Create a freestyle job named "Daily Dose of Satisfaction" to comfort you when you need it. Make it so that it:

- ✓ Has a `NAME` string parameter.
- ✓ When executed, in **separate** shell script steps:
 - displays "Hello dear user!" using `echo` (without any option) and replacing `user` by the value of the `NAME` parameter;
 - executes the `date` command;
 - displays "This is your DDoS number 42." using `echo` (without any option) and replacing `42` by the build number.



For the love of the chocolatinés, please do not use ugly `echo` syntax with tabulations.



Is something missing on your instance?
Try to go to the Update Centre to install relevant plug-ins!



Time to automate... the automation

You now know how to operate Jenkins to do essential work.

However, if you want to set up another instance with the same configuration, you would have to do all the previous steps over again.

That is not cool enough for us and we are too lazy for that.

So it is now time to automate the configuration of Jenkins!

Configuration as Code

The best way to set a configuration in stone is by using a configuration file. And thankfully, a Jenkins plug-in offers the management of its configuration as a config file; its name is... [Configuration as Code \(a.k.a. JCasC\)](#) (what a surprise).

It allows you to define the entire configuration of your instance in just a single YAML file, how cool! Now this is lazy enough for us.

Create a file named `configuration.yml` at the root of your repository which contains the necessary definitions in order to have the same instance as the one manually configured in step 1.



It is possible to have an idea of the current code configuration of your instance by going to the associated page in the settings.

However, do not blindly copy and paste the shown YAML code into your file, as the given code might be incomplete or not completely deterministic when fed back into your instance.



Job DSL

Not all the operations can be achieved with bare YAML and Configuration as Code.

Another plug-in named [Job Domain Specific Language \(a.k.a. Job DSL\)](#) allows you to write scripts for the creation of, well... jobs (yes, everything is obvious there). Define in a `job_dsl.groovy` file the two jobs you created in step 1 using Job DSL, and execute this file using an appropriate instruction in the `configuration.yml` file.



All your DSL scripts must be centralized into one `job_dsl.groovy` file located at the root of the repository. Other Job DSL sources will not be evaluated.

If you use unnecessary plug-ins, the entire DSL correction will fail. You have been warned.

Installed plugins are `configuration-as-code`, `instance-identity`, `job-dsl`, `script-security`, and `structs`.

Feed The Beast

Now that you have a great configuration file, try to reset your Jenkins instance and set it up using only your `configuration.yml` and `job_dsl.groovy` files.



Tired of having to go through the setup wizard each time you restart your instance?

Try to find the environment variable that let you do that.

If you can observe in the new instance the same configuration as the one you manually did in step 1, you won the game! You now have everything you need to tackle the project like the master you are now. Good luck!



v 1.4

{EPITECH}